index that corresponds to a task T, in order to determine if T has initialized C. A null reference at that initialized entry indicates that T has not initialized C, and the system proceeds to the initialization of C by T.

[0020] In one embodiment of the present invention, upon completion of the initialization of a class C by a task T, the system sets an initialized entry of the task class mirror table associated with C to the task class mirror object that holds the representation of C that is private to T. The index to the initialized entry is computed from T's identifier.

[0021] In one embodiment of the present invention, the resolved entry of a task class mirror table is used in cases where testing for class initialization is unneeded but access to the task private part of a class is required (for example, in the case of the Java programming language, accesses to the static variables of a class don't need to be preceded with a class initialization barrier when executing the static initializer or any method of that class; also, the dynamic compiler of the multitasking virtual machine may determine, using some code analysis, that a class initialization barrier is unnecessary when accessing a static variable of that class).

BRIEF DESCRIPTION OF THE FIGURES

[0022] FIG. 1 illustrates computing device 100 in accordance with an embodiment of the present invention.

[0023] FIG. 2 illustrates a typical design of a running multitasking virtual machine 102 in accordance with an embodiment of the present invention.

[0024] FIG. 3A illustrates task class mirror table 302 associated with the shared representation of a class C in accordance with an embodiment of the present invention.

[0025] FIG. 3B illustrates task class mirror table 302 after initialization of class C by a task in accordance with an embodiment of the present invention.

[0026] FIG. 3C illustrates an alternate implementation of task class mirror table 302 associated with the shared representation of a class C in accordance with an embodiment of the present invention.

[0027] FIG. 3D illustrates an alternate implementation of task class mirror table 302 after initialization of class C by a task in accordance with an embodiment of the present invention.

[0028] FIG. 3E illustrates task class mirror table 310 associated with the shared representation of an initialization-less class C in accordance with an embodiment of the present invention.

[0029] FIG. 4 is a flowchart illustrating how a multitasking virtual machine implement access to a global variable of a class by one task in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0030] The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

[0031] The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any

device or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs) and DVDs (digital versatile discs or digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

Computing Device

[0032] FIG. 1 illustrates computing device 100 in accordance with an embodiment of the present invention. Computing device 100 can generally include any type of computer system, including, but not limited to, a computer system based on a microprocessor, a mainframe computer, a digital signal processor, a portable computing device, a personal organizer, a device controller, and a computational engine within an appliance. Computing device 100 includes multitasking virtual machine 102.

[0033] Multitasking virtual machine 102 executes platform-independent code on behalf of multiple tasks such that each task is provided with the illusion that it is the only task being executed. Multitasking virtual machine 102 includes tasks 106, 108, and 110 and shared runtime system 112. Note that multitasking virtual machine 102 can include more or less tasks than shown and described herein. Tasks 106, 108, and 110 and shared runtime system 112 cooperate to execute tasks 106, 108, and 110 as described below in conjunction with FIG. 2.

Multitasking Virtual Machine

[0034] FIG. 2 illustrates a typical design of a running multitasking virtual machine 102 in accordance with an embodiment of the present invention. In FIG.